Adv. Radio Sci., 6, 233–237, 2008
www.adv-radio-sci.net/6/233/2008/

Advances in Radio Science

Fast evaluation of nonlinear functions using FPGAs

S. Lachowicz[1] **and H.-J. Pfleiderer**[2]

[1]Edith Cowan University, Perth, WA, Australia
[2]Ulm University, Ulm, Germany

Abstract. The paper presents a novel method of evaluating the square root function in FPGA. The method uses a linear approximation subsystem with a reduced size of a look-up table. The reduction in the size of the lookup table is twofold. Firstly, a simple linear approximation subsystem uses the lookup table only for the node points. Secondly, a concept of a variable step look-up table is introduced, where the node points are not uniformly spaced, but the spacing is determined by how close to the linear function the approximated function is. The proposed method of evaluating non-linear function and specifically the square root function is practical for word lengths of up to 24 bits. The evaluation is performed in one clock cycle.

1 Introduction

Field Programmable Gate Arrays (FPGA) are becoming widely used in Digital Signal Processing (DSP) systems. Modern FPGAs exhibit very high performance, very large number of IOs and high capacity memory on the chip. The development is rapid and risks are low since the IP can be easily modified. For the low to medium production volume it is not economical to develop an ASIC and, if needed, there are companies like Altera who offer a convenient solution of migration from FPGA to ASIC.

The evolution of FPGAs has enabled to implement more and more sophisticated DSP subsystems in the form of dedicated hardware on the device. In late 1990s FPGAs usually had just a few hardware multipliers per chip. In early 2000s, hardwired multipliers in large numbers (up to $\sim$500) with clock speeds of up to 100 MHz became available. Mid 2000s saw the DSP algorithms like full FIR filters implemented, and currently, powerful arithmetic, such as fast square root and divide, floating point operations and many more are being implemented.

In DSP systems nonlinear functions, such as trigonometric, square root or Bessel functions are often encountered. There are several solutions to this problem, for instance, using a CORDIC core, or look-up tables. A lot of effort is made to reduce the size of the look-up tables because it can become prohibitive for high word lengths. For some functions an existing approach is to create hierarchical lookup tables. In this work we present a different approach to reducing the look-up table size. It is proposed to introduce a variable step of the look-up table, so that the number of entries is higher for more nonlinear segments of the curve and smaller for the parts closer to a linear function. The step length is always a power of 2, hence, 8, 16, 32, etc.

The remainder of the paper is organised as follows. Section 2 presents several methods of computation of nonlinear functions. In Sect. 3 the method of linear approximation is described in more detail. Section 4 deals with the non-uniform step lookup table generation and the associated hardware, and presents the experimental results of square root function implementation. It is followed by conclusions.

2 Nonlinear function computation

2.1 Look-up tables

In general the look-up tables are used for single precision (Hassler, 1995; Sarma, 1995; Wong, 1995; Ercegovac, 2000). For longer word lengths the size of the table becomes prohibitively large. Sometimes the look-up tables are used with conjunction of linear approximation system which reduces the size of the table considerably. When used, the look-up table method is the fastest, and if the higher precision is required it is used as an initial approximation for iterative methods.

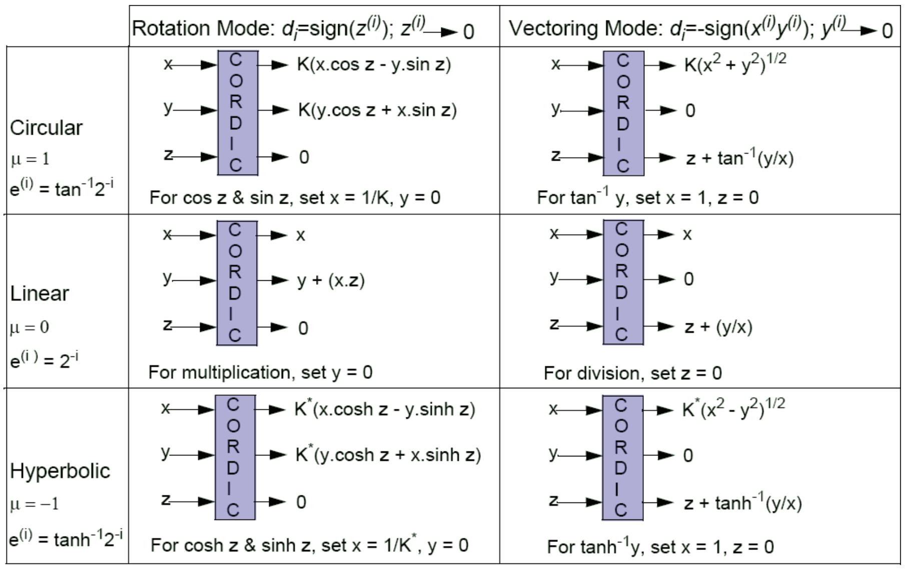
2.2 CORDIC core

COordinate Rotation DIgital Computer (CORDIC) is a very popular IP core used for various computations (Volder, 1959). It is based on the iterative rotation of a vector in Cartesian coordinate system (Givens rotations):

$$x^{(2)} = \cos\theta \left(x^{(1)} - y^{(1)} \tan\theta \right), \tag{1a}$$

Correspondence to: S. Lachowicz
(s.lachowicz@ecu.edu.au)

Table 1. CORDIC functions (after DSPedia).



	Rotation Mode: d_i=sign($z^{(i)}$); $z^{(i)} \to 0$	Vectoring Mode: d_i=-sign($x^{(i)}y^{(i)}$); $y^{(i)} \to 0$
Circular $\mu = 1$ $e^{(i)} = \tan^{-1}2^{-i}$	$x \to$ K(x.cos z - y.sin z); $y \to$ K(y.cos z + x.sin z); $z \to$ 0. For cos z & sin z, set x = 1/K, y = 0	$x \to$ $K(x^2 + y^2)^{1/2}$; $y \to$ 0; $z \to$ $z + \tan^{-1}(y/x)$. For $\tan^{-1}$ y, set x = 1, z = 0
Linear $\mu = 0$ $e^{(i)} = 2^{-i}$	$x \to$ x; $y \to$ y + (x.z); $z \to$ 0. For multiplication, set y = 0	$x \to$ x; $y \to$ 0; $z \to$ z + (y/x). For division, set z = 0
Hyperbolic $\mu = -1$ $e^{(i)} = \tanh^{-1}2^{-i}$	$x \to$ K^*(x.cosh z - y.sinh z); $y \to$ K^*(y.cosh z + x.sinh z); $z \to$ 0. For cosh z & sinh z, set x = 1/K^*, y = 0	$x \to$ $K^*(x^2 - y^2)^{1/2}$; $y \to$ 0; $z \to$ $z + \tanh^{-1}(y/x)$. For $\tanh^{-1}$y, set x = 1, z = 0

Table 2. CORDIC accuracy (after DSPedia).

	Predicted			Simulated		
n/b	8	9	10	8	9	10
3	5.09	5.31	5.43	5.32	5.71	5.80
4	6.03	6.59	6.98	6.22	6.88	7.34
5	6.28	7.13	7.88	6.52	7.56	8.27
6	6.21	7.17	8.10	6.55	7.11	8.12
7	6.06	7.05	8.04	6.42	7.29	8.29
8	5.92	6.91	7.91	6.33	7.29	8.31
9	5.78	6.78	7.78	6.00	7.00	8.00

$$y^{(2)} = \cos\theta \left(y^{(1)} - x^{(1)} \tan\theta \right). \quad \text{(1b)}$$

CORDIC core is widely offered as an IP by the FPGA vendors such as XILINX. Table 1 summarises the functions which can be computed by CORDIC.

As the CORDIC is an iterative method, it requires many clock cycles to achieve the required accuracy. Table 2 presents the accuracy depending on n – number of iterations and b – the number of bits used in the computations.

Also, in terms of used resources the number of slices used by the CORDIC in an FPGA chip is quite considerable.

2.3 Series expansion

Series expansion is an attractive option of calculating a nonlinear function (Ercegovac, 2000). Some of the more useful functions are shown below:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + ..., \quad \text{(2a)}$$

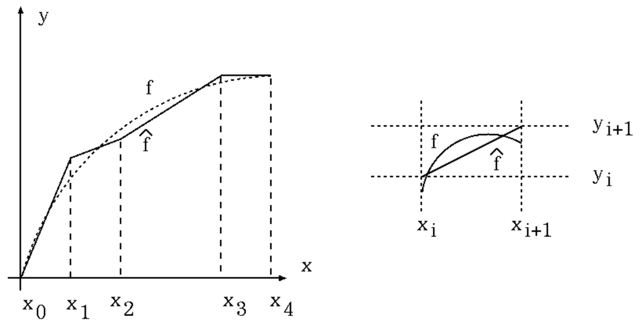


Fig. 1. Piecewise linear approximation of a nonlinear function.

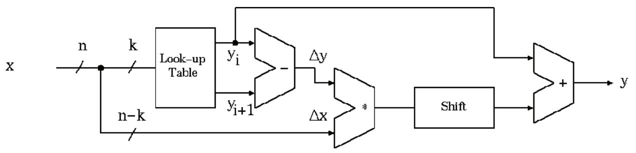


Fig. 2. Hardware for linear function approximation with a fixed step of a look-up table.

$$\cos x = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + ..., \quad \text{(2b)}$$

$$\sqrt{x+1} = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3 - .... \quad \text{(2c)}$$

The method is not iterative, therefore, in principle, the implementation does not have to be pipelined. However, usually the number of required multipliers is high, especially for higher accuracy. Also, preprocessing is generally required to make the argument close to 1. It is worth to be noted that

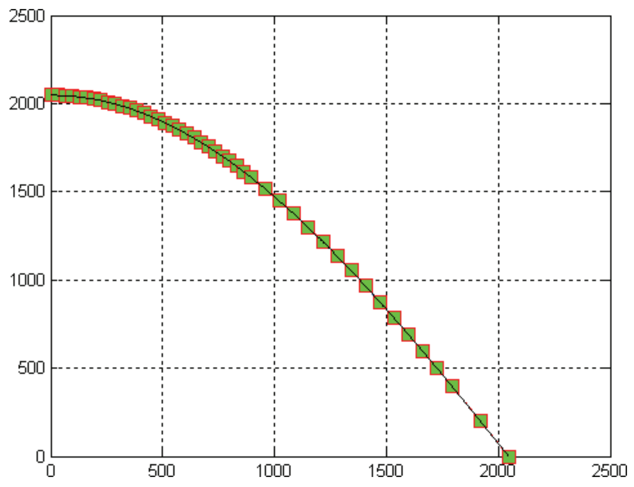


Fig. 3. Node points of the look-up table for the cosine function.

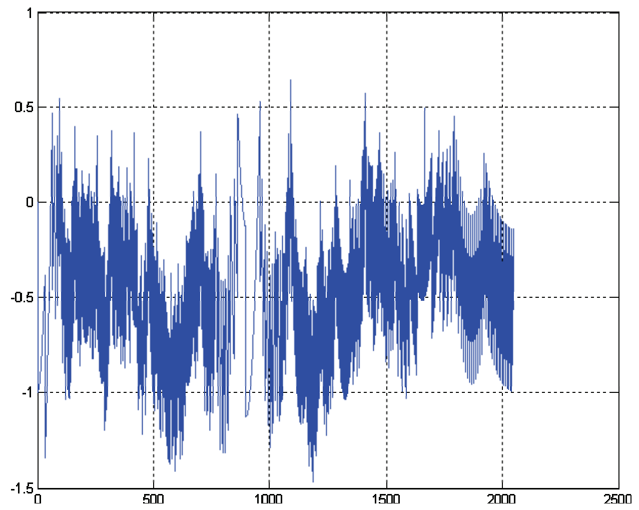


Fig. 4. Approximation error for the cosine function.

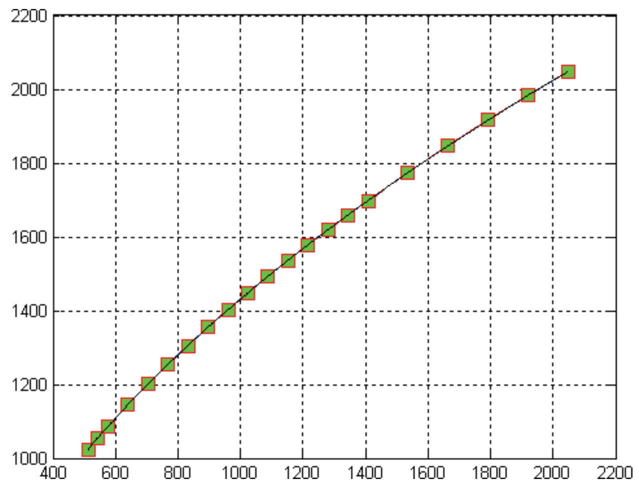


Fig. 5. Node points of an 11-bit look-up table for the sqrt function.

many multiplications do not need to be "large", i.e. 53×53 for double precision. Rectangular multipliers can be used where appropriate. After some preprocessing most of the multiplications are $n/2 \times n/4$, where n is the total number of bits.

2.4 Newton-Raphson iteration

Newton-Raphson iteration is a rapid convergence method often used for functions such as $1/x$ and $1/\mathrm{sqrt}(x)$. The general form is:

$$q_{n+1} = q_n - \frac{f(q_n)}{f'(q_n)}. \tag{3a}$$

The following formulae are applicable to $1/x$ and $1/\mathrm{sqrt}(x)$, respectively:

$$q_{n+1} = q_n(2 - xq_n), \tag{3b}$$

$$q_{n+1} = \frac{1}{2} q_n (3 - xq_n^2). \tag{3c}$$

The method is characterised by quadratic convergence and the number of significant bits roughly doubles with each iteration.

3 Linear approximation with look-up tables

One method of reducing the size of the look-up table is to use a linear approximation circuit. Every function can be approximated by a piecewise linear function as presented in Fig. 1.

Between the node points the function can be expressed using the line equation:

$$\widehat{f}(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i). \tag{4}$$

The hardware mapping is presented in Fig. 2. Here the steps of the look-up table are evenly distributed, so the input word x is split into two parts: the more significant part is an address for the look-up table and the less significant part is the increment of x within the current step of the look-up table. The node values y_i and y_{i+1} are read from the look-up table, the increment Δy is calculated and is subsequently multiplied by the increment Δx. The shift replaces the division as the difference $x_{i+1} - x_i$ is always constant and equal to the number being a power of two.

4 Minimum size look-up table generation

Nonlinear functions usually have a different degree of nonlinearity throughout the range of interest. Therefore, to achieve a particular accuracy while using the linear

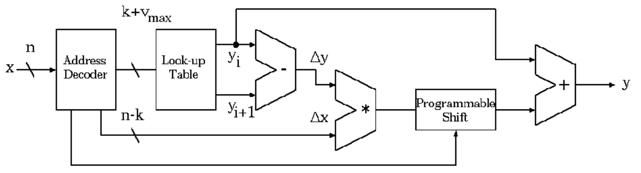


Fig. 6. Hardware for a linear function approximation with a variable step look-up table.

approximation, the distance between the node points in the look-up table can be varied. Where the function is more "linear", the distance between the node points can be increased. In the case of the fixed step table, the step corresponding to the most "non-linear" part of the function has to be maintained throughout the whole look-up table.

Figure 3 presents an example of a look-up table generated for the cosine function, in 11-bit resolution and with the accuracy of ± 1 bit. It can be observed that where the function exhibits higher degree of nonlinearity, the step of the look-up table is reduced.

The look-up table is generated as follows:

1. Each step in the look-up table must be of the length equal to the power of 2.
2. The predetermined accuracy of the linear approximation within one step range is verified for the initial value (large) of the step.
3. If the accuracy is not met then the step is reduced by half and the verification is repeated.
4. If the accuracy criterion is met, the algorithm moves to the next step.
5. The next step can be the same length, shorter or longer, however the longer step can only be made if the remaining part of the range is divisible by the new value of the step length. This ensures that the x value can be split into the look-up table address part and the increment part.

As an example of accuracy of the proposed method of a look-up table generation, let us consider a cosine function from Fig. 3. The number of entries in the look-up table is 45. The accuracy of the approximation has been compared with the full look-up table of 2048 entries. The RMS error in respect to the "accurate" (floating point) curve is 0.35 LSB compared with the RMS error of the full look-up table of 0.28 LSB. Figure 4 shows the actual approximation error throughout the input range.

Square root is an important function often encountered in DSP. Figure 5 presents the look-up table for 11-bits resolution with ± 1 bit accuracy. The look-up table has 21 entries, and for 10-bits the linear approximation circuit generates an

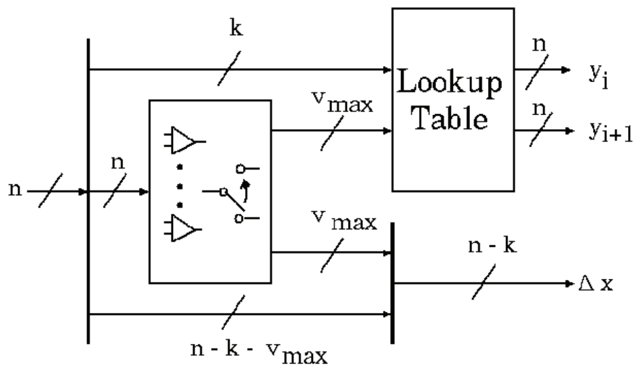


Fig. 7. The address decoder for the circuit from Fig. 6.

exact result. Similar table for a 16-bit accuracy has 262 entries. The full look-up tables would require 1024 and 65 536 entries respectively.

The hardware implementing the variable step look-up table linear approximation differs from the fixed-step as it has to split the input word x into variable length address part and the increment Δx and the fixed shift is replaced by programmable shift dependent on the step of the look-up table. The hardware is shown in Fig. 6.

Figure 7 presents the details of the address decoder in the circuit from Fig. 6. An array of comparators determines the current step of the look-up table and splits the input word x into the address part and the increment part. It also controls the number of positions for the shift. The number of comparators in the array is equal to the number of step changes in the look-up table.

Before the input word is presented to the linear approximation circuit the number has to be normalised. For the square root the input number should be within the range $\{1 \ldots 4\}$ for the output within the range $\{1 \ldots 2\}$. This operation could be referred to as an internally floating-point operation.

As an example of application of the linear approximation circuit with variable step look-up table, a 16-bit square root $\sqrt{x^2+y^2+z^2}$has been implemented in Spartan 3/1000. The resources used are 242 slices (3%), 1 block RAM (4%) and 4 hardware multipliers (16%). Three of the multipliers are used to calculate the squares and one in the linear approximation circuit. The on-chip hardware multipliers are 18×18, and for the linear approximation circuit a 9×9 multiplier would be sufficient. Therefore, if the variable word length multiplier was available (Pfaender, 2007), the resources could be further reduced. The same circuit was also implemented in Virtex 2 FPGA and the operation was verified using ChipScope. The circuit exhibits latency of approximately 11 ns.

5 Conclusions

A novel method of nonlinear function computation in FPGA has been developed. The standard linear approximation circuit has been modified in order to use a look-up table with variable step. This allows to minimise the size of the look-up table and therefore the resources needed for the implementation. The method can be used directly for word lengths up to about 24 bits, and can also be used as an initial value generator for Newton-Raphson iterations if a higher accuracy is required.

Acknowledgements. The support of the German Academic Exchange Office (DAAD) and Ulm University is gratefully acknowledged.

References

Ercegovac, M. D., Lang, T., Muller, J.-M., and Tisserand, A.: Reciprocation, Square Root, Inverse Square Root, and Some Elementary Functions Using Small Multipliers, IEEE Trans. Computers, 49(7), 628–637, 2000.

Hassler, H. and Takagi, N.: Function Evaluation by Table Look-Up and Addition, in: Proc. 12th IEEE Symp. Computer Arithmetic, edited by: Knowles, S. and McAllister, W., 10–16, July 1995.

Pfaender, O. A., Nopper, R., Pfleiderer, H.-J., Zhou, S., and Bermak, A.: Comparison of reconfigurable structures for flexible word-length multiplication, Adv. Radio Sci., 6, 2008.

Sarma, D. D. and Matula, D. W.: Faithful Bipartite ROM Reciprocal Tables, in: Proc. 12th IEEE Symp. Computer Arithmetic, edited by: Knowles, S. and McAllister, W., 17–28, July 1995.

Volder, J.: The cordic trigonometric computing technique, IRE Trans. Electron. Comput., 3, 330–334, 1959.

Wong, W. F. and Goto, E.: Fast Evaluation of the Elementary Functions in Single Precision, IEEE Trans. Computers, 44(3), 453–457, March 1995.